ELSEVIER

Theoretical Computer Science 181 (1997) 141–157

Theoretical Computer Science

Pushdown automata with bounded nondeterminism and bounded ambiguity

Christian Herzog*

Fachbereich Informatik, J.W. Goethe-Universität, Postfach 111932, D-60054 Frankfurt am Main, Germany

Communicated by E. Goles

Abstract

Measures for the amount of ambiguity and nondeterminism in pushdown automata (PDA) are introduced. For every finite k, PDAs with ambiguity at most k are shown to accept exactly the class of languages generated by context-free grammars with ambiguity at most k. PDAs with an amount of nondeterminism at most k accept exactly the class of the unions of k deterministic context-free languages. For all finite or infinite k, k' with $k \leqslant k'$ there is a language that can be accepted by a PDA with ambiguity k and nondeterminism k' but by no PDA with less ambiguity or less nondeterminism. For every finite k, it is shown that the tradeoff from a description by a PDA with ambiguity $k+1$ and nondeterminism $k+1$ to PDAs with ambiguity k is bounded by no recursive function. The tradeoff from PDAs with ambiguity 1 and nondeterminism $k+1$ to PDAs with nondeterminism k also is bounded by no recursive function. The tradeoff from PDAs with branching k to PDAs with ambiguity k and branching k is at most exponential.

1. Introduction

The concepts of ambiguity and nondeterminism play a fundamental role in automata theory, and there are some famous open problems regarding nondeterminism, for example, the $\mathscr{P}$ versus $\mathscr{NP}$ or the DLBA versus NLBA problem. For the pushdown automata (PDA) model, many questions concerning ambiguity and nondeterminism already have been answered, but some problems worth addressing still remain. Forbidding ambiguity or nondeterminism in PDAs has advantages for some applications of PDAs, for example, for the syntax checking of programming languages. On the other hand, this restriction has the disadvantage that not every context-free language can be described by an unambiguous or deterministic PDA. Second, this restriction may change complexity: The savings in size achieved by ambiguous over unambiguous or by nondeterministic over deterministic PDAs are not bounded by any recursive function.

* E-mail: herzog@psc.informatik.uni-frankfurt.de.

PII S0304-3975(96)00267-8

We want to examine the situation between completely forbidding ambiguity or nondeterminism and allowing an arbitrary amount of ambiguity or of nondeterminism. Therefore, we introduce measures for the amount of ambiguity and of nondeterminism in PDAs. In particular, we are interested in PDAs where this amount is bounded by finite constants. We then study the corresponding classes of languages and the savings in complexity obtained.

2. Preliminaries

We use the following notations and definitions of automata and grammars as introduced in [5]:

Definition 1. A *pushdown automaton* (PDA) accepting by final states is a tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, where Q is the finite set of states, Σ is the input alphabet, Γ is the stack alphabet, q_0 is the initial state, Z_0 the initial stack symbol, $F \subseteq Q$ the set of accepting states and δ is the set of transition rules and a mapping from $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma$ into finite subsets of $Q \times \Gamma^*$.

A configuration of M is $c = (q, x, \omega) \in Q \times \Sigma^* \times \Gamma^*$, where q is the actual state, x the remaining part of the input word and ω the actual contents of the stack with the topmost symbol left. The rules of M induce a move relation $\underset{M}{\vdash}$ between configurations in the usual way. The relation $\overset{*}{\underset{M}{\vdash}}$ the reflexive and transitive closure of $\underset{M}{\vdash}$. The language accepted by M is

$$L(M) = \{x \in \Sigma^* \mid (q_0, x, Z_0) \overset{*}{\underset{M}{\vdash}} (p, \varepsilon, \omega) \text{ for some } p \in F, \omega \in \Gamma^*\}.$$

A PDA is called deterministic PDA (DPDA), if for every configuration there is at most one possible next configuration, that is, if

1. For all $q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$ and $Z \in \Gamma$, $\delta(q, a, Z)$ has at most one element.
2. For all $q \in Q$, $Z \in \Gamma$ with $\delta(q, \varepsilon, Z) \neq \emptyset$, $\delta(q, a, Z) = \emptyset$ holds for all $a \in \Sigma$.

The size $\|M\|$ of a PDA M is the total number of symbols in its transition rules. The class of languages accepted by PDAs and DPDAs is called the class of context-free languages (CFL) and deterministic context-free languages (DCFL), respectively.

Definition 2. A *context-free grammar* (CFG) is a tuple $G = (V, \Sigma, P, S)$ where V is the finite set of symbols, $\Sigma \subseteq V$ is the set of terminals, P is the finite set of productions, and $S \in V - \Sigma$ is the start symbol. The productions induce a derivation relation $\Rightarrow$ between words in V^* in the usual way. The relation $\overset{*}{\Rightarrow}$ is the reflexive and transitive closure of $\Rightarrow$. A sequence of derivation steps is called leftmost if in every step a production is applied to the leftmost nonterminal. The language generated by G is

$$L(G) = \{x \in \Sigma^* \mid S \overset{*}{\Rightarrow} x\}.$$

The size $\|G\|$ of a CFG G is the total number of symbols in its productions.

Definition 3. A *Turing-machine* (TM) with one tape infinite to the right is a tuple $M = (Q, \Sigma, \Gamma, \not\emptyset, \delta, q_0, F)$ where Q is the finite set of states, Σ is the input alphabet, $\Gamma \supseteq \Sigma$ is the finite set of tape symbols, $\not\emptyset \in \Gamma - \Sigma$ is the blank symbol, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of accepting (or halting) states and δ is the set of transition rules and a mapping from $(Q-F)\times\Gamma$ into subsets of $Q\times(\Gamma-\{\not\emptyset\})\times\{L,R\}$. Note that we neither allow transition rules from accepting states nor rules writing the blank symbol.

A configuration of M is $c = \omega_1 q \omega_2 \in \Gamma^* Q \Gamma^*$ with q the actual state and $\omega_1\omega_2$ the nonblank portion of the tape. The symbol under the head of the TM is the first symbol of ω_2 or the blank symbol, if $\omega_2 = \varepsilon$. The set of transition rules induces a relation $\underset{M}{\vdash}$ between configurations of M in the usual way, the relation $\overset{*}{\underset{M}{\vdash}}$ is the reflexive and transitive closure of $\underset{M}{\vdash}$. The language accepted by M is

$$L(M) = \{x \in \Sigma^* \mid q_0 x \overset{*}{\underset{M}{\vdash}} \omega_1 p \omega_2 \text{ for some } p \in F, \omega_1, \omega_2 \in \Gamma^*\}.$$

A TM is called deterministic TM, if for every configuration there is at most one possible next configuration, that is, if $\delta(q,X)$ never contains more than one element. The size $\|M\|$ of a TM M is the total number of symbols in its transition rules.

Finally, we cite two pumping lemmas from [3], one for CFLs and one for DCFLs:

Lemma 4 (Ogden-Lemma for CFLs, Harrison [3, Lemma 6.2.1]). *Let $G = (V, \Sigma, P, S)$ be a CFG. Then there is a constant $C = C(G)$ for G, such that for every word $w \in L(G)$ with at least C marked positions, there is a factorization $w = v_1 v_2 v_3 v_4 v_5$ with:*

(1) *v_1, v_2 and v_3 all have marked positions or v_3, v_4 and v_5 all have marked positions.*

(2) *$v_2 v_3 v_4$ has at most C marked positions.*

(3) *There is an $A \in V - \Sigma$ such that for all $n \geqslant 0$,*

$$S \overset{*}{\Rightarrow} v_1 A v_5 \overset{*}{\Rightarrow} v_1 v_2^n A v_4^n v_5 \overset{*}{\Rightarrow} v_1 v_2^n v_3 v_4^n v_5 \in L(G).$$

Moreover, there is a recursive function φ, such that the constant $C(G) = \varphi(\|G\|)$ is a valid pumping constant for the grammar G.

Lemma 5 (Ogden-Lemma for DCFLs, Harrison [3, Lemma 11.8.3]). *Let L be a DCFL over the alphabet Σ. Then there is a constant $C = C(L)$ for L, such that for every word $w \in L$ with at least C marked positions, there is a factorization $w = v_1 v_2 v_3 v_4 v_5$ with:*

(1) *$v_2 \neq \varepsilon$.*

(2) *For all $n \geqslant 0$, $v_1 v_2^n v_3 v_4^n v_5 \in L$.*

(3) *v_1, v_2 and v_3 all have marked positions or v_3, v_4 and v_5 all have marked positions.*

(4) *$v_2 v_3 v_4$ has at most C marked positions.*

(5) *If* $v_5 \neq \varepsilon$, *then for all* $m, n \geqslant 0$ *and all* $u \in \Sigma^*$,

$$v_1 v_2^{m+n} v_3 v_4^n u \in L \Leftrightarrow v_1 v_2^m v_3 u \in L.$$

Moreover, there is a recursive function φ, *such that for every DPDA* M *the constant* $C(L(M)) = \varphi(\|M\|)$ *is a valid pumping constant for the language* $L(M)$.

3. Ambiguity

Measuring the amount of ambiguity in context-free grammars is well known, see, for example, [3] or [8]. We transfer this measure to PDAs in such a way that the classes of languages described by PDAs and CFGs with the same amount of ambiguity are equivalent.

Definition 6. Like in [3, Section 7.3], let the *ambiguity* $\alpha_G(x)$ of a word x in a CFG G and the ambiguity α_G of G be defined as

$$\alpha_G(x) = \text{number of leftmost derivations of } x \text{ in } G,$$

$$\alpha_G = \sup\{\alpha_G(x) \mid x \in L(G)\}.$$

Analogously, let the ambiguity α_M of a PDA M be

$$\alpha_M(x) = \text{number of accepting computations on } x \text{ in } M,$$

$$\alpha_M = \sup\{\alpha_M(x) \mid x \in L(M)\}.$$

For all $k \in \mathbb{N} \cup \{\infty\}$ let

$$\text{CFG}(\alpha \leqslant k) = \{G \mid G \text{ CFG with } \alpha_G \leqslant k\},$$

$$\text{PDA}(\alpha \leqslant k) = \{M \mid M \text{ PDA with } \alpha_M \leqslant k\},$$

$$\text{CFL}(\alpha \leqslant k) = \{L(M) \mid M \in \text{PDA}(\alpha_M \leqslant k)\}.$$

For the special case $k = 1$, this definition contains the usual definition of unambiguity, that is, $\text{CFG}(\alpha \leqslant 1) = \text{UCFG}$, $\text{PDA}(\alpha \leqslant 1) = \text{UPDA}$, and $\text{CFL}(\alpha \leqslant 1) = \text{UCFL}$. The class $\text{CFL}(\alpha \leqslant k)$ was defined by PDAs rather than by CFGs, but the following theorem will show that this definition would have been equivalent:

Theorem 7. *For all* $k \in \mathbb{N} \cup \{\infty\}$,

$$\{L(M) \mid M \in \text{PDA}(\alpha \leqslant k)\} = \{L(G) \mid G \in \text{CFG}(\alpha \leqslant k)\}.$$

Proof. It is well known that for every CFG G we can construct an equivalent PDA M with final states and vice versa, see, for example, [5, Section 5.3]. Since for every leftmost derivation of a word in the CFG G there is exactly one corresponding accepting computation of the PDA M, the constructions from [5] preserve the degree of ambiguity, that is, $\alpha_M(x) = \alpha_G(x)$ for all words x and therefore, $\alpha_M = \alpha_G$. □

4. Nondeterminism

There are several measures for the amount of nondeterminism in a PDA, for example, in [13] (with corrections in [9]) or recently in [10], where a so called "minmax" measure is defined. The minmax measure of an input word is the minimal number of nondeterministic moves necessary to accept this word. The minmax measure of a PDA is the supremum of the minmax measures of all words accepted by this PDA. We know from [10] that the class of PDAs with minmax measure zero accepts exactly the class of DCFLs, the class with minmax measure one or equivalently with finite measure accepts finite unions of DCFLs, and the class with infinite measure accepts general CFLs.

We introduce a new measure called the branching of a PDA, which is similar to the minmax measure, but also considers the multiplicities of the nondeterministic moves counted in the minmax measure. Contrary to the two levels in the hierarchy of classes of languages accepted by PDAs with finite minmax measure, our measure will yield an infinite hierarchy. Our definition of the branching of a PDA is the analog of a corresponding measure for finite automata from [2].

Definition 8. Let the *branching* β_M of a PDA M be defined the following way: The branching of a single move of M is the number of next configurations that are possible from the given configuration. The branching $\beta_M(\pi)$ of a computation π is the product of the branchings of all the moves in this computation and

$$\beta_M(x) = \min\{\beta_M(\pi) \mid \pi \text{ is acc. comp. of } M \text{ on } x\} \quad \text{for words } x \in L(M),$$

$$\beta_M = \sup\{\beta_M(x) \mid x \in L(M)\}.$$

Moreover, for all $k \in \mathbb{N} \cup \{\infty\}$ let

$$\mathrm{PDA}(\beta \leqslant k) = \{M \mid M \text{ PDA with } \beta_M \leqslant k\},$$

$$\mathrm{CFL}(\beta \leqslant k) = \{L(M) \mid M \in \mathrm{PDA}(\beta \leqslant k)\}.$$

Fig. 1 shows the example of a computation tree of some PDA and the branchings of the accepting computations π_1 and π_2 on an input word x with $\beta_M(x) = 8$.

In a PDA M with finite branching $\beta_M \leqslant k$, for every word in $L(M)$, there is an accepting computation with branching at most k. Thus, all computations of M can

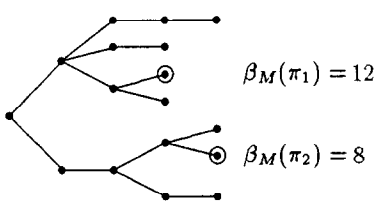

Fig.1. Computations on a word x with $\beta_M(x) = 8$.

be cut off at that point where the branching of the moves made so far exceeds the value k, without changing the language accepted. So the branching of a PDA tells us, up to which width the computation tree of some input word has to be examined until an accepting computation is found. One could also say that the branching of a PDA reflects the amount of parallelism needed for a deterministic real-time simulation of this PDA, that is, the number of copies that have to be created during a simulation. This fact is one of the reasons why we chose this measure for the amount of nondeterminism.

Note that the branching of a word is not necessarily equal to the number of computations on this word, which may be as well greater as less than the branching.

Note also that in a PDA, where every move has branching at most two, the logarithm of the branching is exactly the minmax measure of this PDA. In this case the two measures are essentially equivalent because they only use different units of measure.

There is no relation between the ambiguity and the branching of a PDA, since there is as well a PDA M with $\alpha_M = 1$ and $\beta_M = \infty$, as there is a PDA M with $\alpha_M = \infty$ and $\beta_M = 1$.

Contrary to the fact that a UPDA is the same as a PDA with ambiguity one, a DPDA is not the same as a PDA with branching one because a DPDA has no nondeterministic transition rules, whereas a PDA with branching one may have some but just does not use them in accepting computations. However, at least the classes of languages, DCFL and $\mathrm{CFL}(\beta \leqslant 1)$, are equal. This will be shown as a special case of the following theorem, which says that the class of languages accepted by PDAs with branching k is the class of unions of k DCFLs.

Theorem 9. *For all* $k \in \mathbb{N}$,

$$\mathrm{CFL}(\beta \leqslant k) = \bigsqcup^{k} \mathrm{DCFL}.$$

Proof. The inclusion $\bigsqcup^k \mathrm{DCFL} \subseteq \mathrm{CFL}(\beta \leqslant k)$ is obvious, since any k DPDAs $D_1, \ldots, D_k$ can be unified to a PDA M with $L(M) = \bigcup_{i=1}^{k} L(D_i)$ and $\beta_M \leqslant k$ by introducing a new initial state of M and by branching from this state to the initial configurations of all the k DPDAs.

To prove the other inclusion $\mathrm{CFL}(\beta \leqslant k) \subseteq \bigsqcup^k \mathrm{DCFL}$ we show in three steps that for every $M \in \mathrm{PDA}(\beta \leqslant k)$ there are k DPDAs $D_1, \ldots, D_k$, such that $\bigcup_{i=1}^{k} L(D_i) = L(M)$:

1. For every $M \in \mathrm{PDA}(\beta \leqslant k)$ there is an equivalent $M' \in \mathrm{PDA}(\beta \leqslant k)$ that accepts every word immediately upon consuming the last input symbol, that is, without subsequent ε-moves.
2. For every such $M' \in \mathrm{PDA}(\beta \leqslant k)$ there is an equivalent $M'' \in \mathrm{PDA}(\beta \leqslant k)$ in which from no configuration an ε- and also a Σ-move is possible and which hence is called a PDA without ε-Σ-nondeterminism.
3. For every such $M'' \in \mathrm{PDA}(\beta \leqslant k)$ there are k DPDAs $D_1, \ldots, D_k$ such that $\bigcup_{i=1}^{k} L(D_i) = L(M'')$.

Proof of 1. We construct the desired PDA M' from M using the predicting machine from [5, Section 10.3] with the predicting language $\{\varepsilon\}$. We omit the formal construction since it is an easy extension of Exercise 10.7 in [5], where the problem is solved for DPDAs instead of PDAs with branching k. The detailed construction can also be found in [4].

Proof of 2. For this M' we have to find an equivalent PDA M'' without ε-Σ-nondeterminism and with $\beta_{M''} \leqslant k$. M'' is constructed from M' by allowing it to store an input symbol in its states. The first move M'' has to make is to read the first input symbol and store it. An ε-move of M' then is simulated by an ε-move of M'' which leaves the stored input symbol unchanged. A Σ-move of M' is simulated by an ε-move of M'' which deletes the stored symbol, followed by a Σ-move of M'' which reads the next input symbol and stores it. The formal construction of M'' for some given $M' = (Q', \Sigma, \Gamma', \delta', q_0', Z_0', F')$ is as follows: $M'' = (Q' \cup (Q' \times \Sigma), \Sigma, \Gamma', \delta'', q_0', Z_0', F')$ with the rules

$$\delta''(q,a,Z) \ni ([q,a],Z) \quad \text{for all } q \in Q',\ a \in \Sigma,\ Z \in \Gamma',$$

$$\delta''([q,a],\varepsilon,Z) \ni ([p,a],\omega) \quad \text{for all } \delta'(q,\varepsilon,Z) \ni (p,\omega),\ a \in \Sigma,$$

$$\delta''([q,a],\varepsilon,Z) \ni (p,\omega) \quad \text{for all } \delta'(q,a,Z) \ni (p,\omega).$$

M'' has no ε-Σ-nondeterminism, because from states with a stored symbol only ε-moves, and from states without one only Σ-moves are possible. For all $a \in \Sigma$ there is a computation

$$(q,a,\omega) \overset{1}{\underset{M''}{\vdash}} ([q,a],\varepsilon,\omega) \overset{t}{\underset{M''}{\vdash}} ([q',a],\varepsilon,\omega') \overset{1}{\underset{M''}{\vdash}} (q'',\varepsilon,\omega'')$$

in M'' if and only if there is a computation

$$(q,a,\omega) \overset{t}{\underset{M'}{\vdash}} (q',a,\omega') \overset{1}{\underset{M'}{\vdash}} (q'',\varepsilon,\omega'')$$

in M'. Those two computations have the same branching because the first move of M'' has branching one and the other moves of M'' have the same branching as the corresponding moves of M'.

Since M' always accepts immediately upon consuming the last input symbol, $x \in L(M')$ if and only if there is an accepting computation of M' on x whose last move is a Σ-move. By concatenating several of the above computations consuming a single symbol $a \in \Sigma$, we find that this is the case if and only if there is an accepting computation of M'' on x. Therefore, $L(M'') = L(M')$. Moreover, we have $\beta_{M''} = \beta_{M'}$ because those two accepting computations have the same branching.

The assumption that M' accepts immediately upon consuming the last symbol was necessary because M'' is not able to simulate ε-moves after the last Σ-move of M'.

Proof of 3. We have to show that for every $M'' \in \mathrm{PDA}(\beta \leqslant k)$ without ε-Σ-nondeterminism, there are k DPDAs $D_1, \ldots, D_k$, such that $\bigcup_{i=1}^{k} L(D_i) = L(M'')$. The basic idea

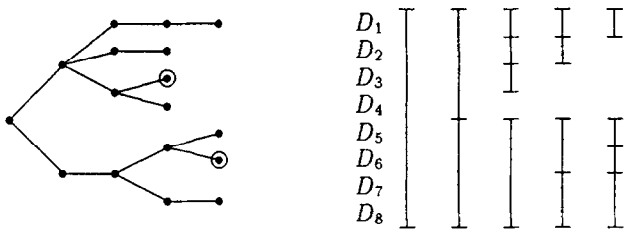

Fig. 2. Computation tree and groups of DPDAs.

of this construction is to let each DPDA simulate exactly one computation of the given PDA M'' with branching at most k, because there are at most k of them, and at least one of them is accepting, if the input word is accepted by M''. This is done the following way: At each moment, there are groups of DPDAs that are in the same configuration. If during the simulation of M'' a DPDA has to simulate a move of M'' with branching, for example, two, then the first half of its group simulates the first possible move, the second half simulates the second move. From now on, both halves of the group form an own group of half the size. If the size of a group is less than the branching of the move to be simulated, a further subdivision of this group is impossible and all the DPDAs in the group have no possible next move. Fig. 2 shows the example of a computation tree of some PDA with branching $k=8$ and the corresponding subdivision of the DPDAs into groups during the simulation. Note that the subdivision of the groups corresponds to the creation of copies in a real-time simulation as mentioned above, and that the size k of the initial group corresponds to the maximum number of copies created. M'' is not allowed to have ε-Σ-nondeterminism, because every DPDA simulating an ε-move must know the branching of this move without knowing the next input symbol, and in PDAs with ε-Σ-nondeterminism, this branching may depend on the next symbol.

For the formal construction of the DPDAs $D_1,\ldots,D_k$, let $M''=(Q'',\Sigma,\Gamma'',\delta'',q_0'',Z_0'',F'')$. To accomplish the desired subdivision into groups, every DPDA stores in its states its own number, the number of the first member of its group and the size of this group. Thus, we set

$$D_i=(Q''\times\{1,\ldots,k\}^3,\Sigma,\Gamma'',\delta,[q_0'',i,1,k],Z_0'',F''\times\{1,\ldots,k\}^3).$$

Let $\delta''(q,a,Z)\ni(p,\omega)$ be a transition rule of M'' and let r be the cardinality of $\delta''(q,a,Z)$. (Note that r is the branching of every move using this rule since M'' has no ε-Σ-nondeterminism.) Then for all $j,n\in\{1,\ldots,k\}$, the DPDA D_i contains the rule

$$\delta([q,i,j,n],a,Z)\ni([p,i,j+l\lfloor n/r\rfloor,\lfloor n/r\rfloor],\omega)$$

if the following conditions hold:

1. $1\leqslant j\leqslant i\leqslant j+n-1\leqslant k$ (D_i is a member of the group of size n starting at D_j)
2. $n\geqslant r$ (the group is large enough for a subdivision into r subgroups)

3. (p,ω) is the $(l+1)$st element of $\delta''(q,a,Z)$, where $l=\lfloor (i-j)/\lfloor n/r\rfloor\rfloor$ (D_i is a member of the $(l+1)$st subgroup and simulates the $(l+1)$st move)

Since replacing every state of the form $[q,i,j,n]$ in an accepting computation of D_i on some word x by the state q yields a corresponding accepting computation of M'' on x, we have $L(D_i)\subseteq L(M'')$ for every i and hence $\bigcup_{i=1}^{k} L(D_i)\subseteq L(M'')$.

To show the other inclusion we use the fact that, if

$$(q,x,\omega)\overset{t}{\underset{M''}{\vdash}}(q',x',\omega')$$

is a computation of M'' with some branching $r\leqslant k$ and if j, n are integers with $r\leqslant n\leqslant k$ and $1\leqslant j\leqslant k-n+1$, then there are at least $\lfloor n/r\rfloor$ many D_i such that

$$([q,i,j,n],x,\omega)\overset{t}{\underset{D_i}{\vdash}}([q',i,j',\lfloor n/r\rfloor],x',\omega')$$

for some j'. This can be proved by induction on the length of the computation.

For every $x\in L(M'')$ there must be some accepting computation in M'' with branching $r\leqslant k$. Applying the above claim to this computation with $j=1$ and $n=k$, we find an accepting computation on x in D_i for at least $\lfloor k/r\rfloor\geqslant 1$ many i, so $x\in\bigcup_{i=1}^{k} L(D_i)$. Therefore, we have shown that $L(M'')\subseteq\bigcup_{i=1}^{k} L(D_i)$, and the proof is complete. □

The meaning of the equality $\mathrm{CFL}(\beta\leqslant k)=\bigsqcup^{k}\mathrm{DCFL}$ in the last theorem is that, regarding the class of describable languages, it is equivalent whether the nondeterminism in a PDA with branching k is distributed over its computations in any way or whether the nondeterminism consists of simply choosing one of k DPDAs. This characterization of the class $\mathrm{CFL}(\beta\leqslant k)$ is another justification for choosing the branching as a measure for the amount of nondeterminism in PDAs.

Note that this equality does not hold for $k=\infty$, since $\mathrm{CFL}(\beta\leqslant\infty)=\mathrm{CFL}$ but every language, whether context-free of not, is the union of infinitely many singleton DCFLs. An easy corollary of the theorem is the inclusion

$$\mathrm{CFL}(\beta\leqslant k)\subseteq\mathrm{CFL}(\alpha\leqslant k).$$

Note that, however, $\mathrm{PDA}(\beta\leqslant k)\not\subseteq\mathrm{PDA}(\alpha\leqslant k)$.

5. A hierarchy of CFLs

It is well known that there are unambiguous CFLs which cannot be accepted by a deterministic PDA, and that there are CFLs which cannot be accepted by an unambiguous PDA. So DCFL is a proper subset of UCFL, and UCFL is a proper subset of CFL.

Obviously, for all $k\in\mathbb{N}$ the inclusions $\mathrm{CFL}(\alpha\leqslant k)\subseteq\mathrm{CFL}(\alpha\leqslant k+1)$ and $\mathrm{CFL}(\beta\leqslant k)\subseteq\mathrm{CFL}(\beta\leqslant k+1)$ hold. Moreover, $\mathrm{CFL}(\beta\leqslant k)$ is a subset of $\mathrm{CFL}(\alpha\leqslant k)$ because of the corollary to Theorem 9. Using languages from [8] and [6], we will prove all these inclusions to be proper.

Lemma 10 (Maurer [8]). *For all $k \in \mathbb{N}$ let*

$$A_k = \bigcup_{i=1}^{k} \{a_1^{n_1} a_2^{n_2} a_3^{n_2} \dots a_{2i-1}^{n_i} a_{2i}^{n_1} a_{2i+1}^{n_{i+1}} \dots a_{2k-1}^{n_k} a_{2k}^{n_k} \mid n_1, \dots, n_k \geqslant 1\},$$

$$A_\infty = A_2^*.$$

Then A_k is inherently ambiguous of degree k and A_∞ is inherently ambiguous of infinite degree, that is,

$$A_k \in \mathrm{CFL}(\alpha \leqslant k), \quad A_k \notin \mathrm{CFL}(\alpha < k),$$

$$A_\infty \in \mathrm{CFL}, \quad A_\infty \notin \mathrm{CFL}(\alpha < \infty).$$

Proof. It is easy to see that $A_k \in \mathrm{CFL}(\alpha \leqslant k)$ and $A_\infty \in \mathrm{CFL}$.

The fact that $A_k \notin \mathrm{CFL}(\alpha < k)$ was already shown in [8]. We reprove this result using Lemma 4. Let $A_k = L(G)$ for some CFG G and let C be the pumping constant for G from Lemma 4. Now consider the words

$$w_i = a_1^C a_2^{C!+C} \dots a_{2i-1}^{C!+C} a_{2i}^C a_{2i+1}^{C!+C} \dots a_{2k}^{C!+C} \quad \text{for } i = 1, \dots, k,$$

where all the a_{2i}'s are marked. Then for every factorization $w = v_1 v_2 v_3 v_4 v_5$ satisfying conditions (1)–(3) of Lemma 4, $v_2 = a_1^l$ and $v_4 = a_{2i}^l$ for some $0 < l < C$. Setting $n = 1 + C!/l$ in (3), we get a derivation

$$S \overset{*}{\Rightarrow} v_1 A v_5 \overset{*}{\Rightarrow} v_1 v_2^n A v_4^n v_5 \overset{*}{\Rightarrow} v_1 v_2^n v_3 v_4^n v_5 = a_1^{C!+C} \dots a_{2k}^{C!+C} = w.$$

Since these k derivations for the word w are distinct for $i = 1, \dots, k$, the CFG G has ambiguity at least k. Thus, there can be no CFG and no PDA for A_k with ambiguity less than k, so $A_k \notin \mathrm{CFL}(\alpha < k)$ holds.

To prove $A_\infty \notin \mathrm{CFL}(\alpha < \infty)$, we assume that $A_\infty = L(G)$ for some $G \in \mathrm{CFG}(\alpha < \infty)$, that is, $G \in \mathrm{CFG}(\alpha < 2^k)$ for some finite k. Let C be the pumping constant for this CFG from Lemma 4. Like above, we can show that there are at least 2^k different derivations for the word

$$w = (a_1^{C!+C} a_2^{C!+C} a_3^{C!+C} a_4^{C!+C})^k.$$

Since this is a contradiction to $G \in \mathrm{CFG}(\alpha < 2^k)$, there can be no CFG and no PDA for A_∞ with finite ambiguity, so $A_\infty \notin \mathrm{CFL}(\alpha < \infty)$ holds. □

Lemma 11 (Kintala [6]). *For all $k \in \mathbb{N}$ let*

$$B_k = \bigcup_{i=1}^{k} \{b_1^n b_2^{in} \mid n \geqslant 1\},$$

$$B_\infty = B_2^*.$$

Then B_k is unambiguous and inherently nondeterministic of degree k and B_∞ is unambiguous and inherently nondeterministic of infinite degree, that is,

$$B_k \in \mathrm{CFL}(\alpha \leqslant 1, \beta \leqslant k), \quad B_k \notin \mathrm{CFL}(\beta < k),$$

$$B_\infty \in \mathrm{CFL}(\alpha \leqslant 1), \quad B_\infty \notin \mathrm{CFL}(\beta < \infty).$$

Proof. It is easy to see that $B_k \in \mathrm{CFL}(\alpha \leqslant 1, \beta \leqslant k)$ and $B_\infty \in \mathrm{CFL}(\alpha \leqslant 1)$.

In [6], it was shown that B_k is not the union of less than k DCFLs and thus not in $\mathrm{CFL}(\beta < k)$. We reprove this result using Lemma 5: Assume B_k were the union of less than k DCFLs. Let C be the maximum of the pumping constants for these DCFLs from Lemma 5. Now consider the words

$$w_i = b_1^C b_2^{iC} \quad \text{for } i = 1, \ldots, k.$$

Since there are k words but less than k DCFLs, one language L must contain two of these words, say w_j and $w_{j'}$ with $j < j'$. We mark all the b_1's in w_j. Then for every factorization $w_j = v_1 v_2 v_3 v_4 v_5$ satisfying the conditions (1)–(5) of the pumping lemma, $v_2 = b_1^l$ and $v_4 = b_2^{jl}$ for some $0 < l < C$. Now we set $m = 1$ and $u = v_4 v_5 b_2^{(j'-j)C}$ in (5). Then $v_1 v_2^m v_3 u = w_{j'}$ is in L and thus for all $n \geqslant 0$,

$$v_1 v_2^{m+n} v_3 v_4^n u = b_1^{C+ln} b_2^{j'C+jln} \in L.$$

Since $0 < l < C$, this only is possible if $j = j'$, contradicting $j < j'$. Therefore, $B_k \notin \mathrm{CFL}(\beta < k)$.

The fact that $B_\infty \notin \mathrm{CFL}(\beta < \infty)$ is proved in a similar way: We assume that $B_\infty \in \mathrm{CFL}(\beta \leqslant k)$ for some finite k, so B_∞ is the union of k DCFLs. Now consider the $k+1$ words

$$w_i = (b_1^C b_2^C)^i (b_1^C b_2^{2C})^{k-i} \quad \text{for } i = 0, \ldots, k,$$

where C is the maximum of the pumping constants for the k DCFLs from Lemma 5. Like above, we then can show that two of these words w_i can never be in the same DCFL. Because of this contradiction, $B_\infty \notin \mathrm{CFL}(\beta < \infty)$ holds. □

Lemma 12. *For all $k, k' \in \mathbb{N} \cup \{\infty\}$ with $k \leqslant k'$ let*

$$L_{k,k'} = A_k \cup B_{k'}.$$

Then $L_{k,k'}$ is inherently ambiguous of degree k and inherently nondeterministic of degree k', that is,

$$L_{k,k'} \in \mathrm{CFL}(\alpha \leqslant k, \beta \leqslant k'), \quad L_{k,k'} \notin \mathrm{CFL}(\alpha < k), \quad L_{k,k'} \notin \mathrm{CFL}(\beta < k').$$

Proof. We can construct a PDA for $L_{k,k'}$ that branches to a PDA for A_k with $\alpha \leqslant k$ and $\beta \leqslant k$, if the first input symbol is an a_1, and that branches to a PDA for $B_{k'}$ with $\alpha \leqslant 1$ and $\beta \leqslant k'$, if the first symbol is a b_1. Then this PDA has ambiguity k and

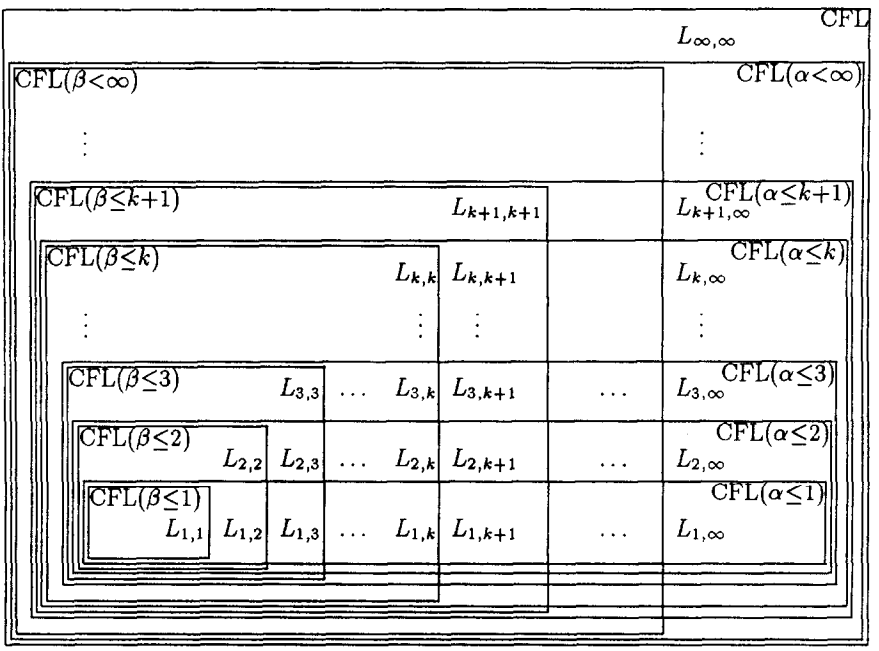

Fig. 3. Hierarchy of CFLs.

branching k', so $L_{k,k'} \in \mathrm{CFL}(\alpha \leqslant k, \beta \leqslant k')$. If there were a PDA for $L_{k,k'}$ with $\alpha < k$, then deleting all the rules consuming input symbols b_1 or b_2 yields a PDA for A_k with $\alpha < k$. Since this is a contradiction to Lemma 10, $L_{k,k'} \notin \mathrm{CFL}(\alpha < k)$. For the same reasons, $L_{k,k'} \notin \mathrm{CFL}(\beta < k')$ holds. □

Using these languages $L_{k,k'}$, we are able to prove the main result of this section: There is an infinite hierarchy of CFLs accepted by PDAs with finite degrees of ambiguity and nondeterminism, as presented in Fig. 3.

This infinite hierarchy means that every time the allowed amount of ambiguity or the allowed amount of nondeterminism in a PDA is increased by the smallest unit possible, that is, either one more accepting computation or one more branch in the computation tree is allowed, then a language becomes describable by such PDAs which was not describable before.

6. Tradeoffs

Finally, we will study the savings in complexity that can be achieved by describing a language by a PDA with a higher amount of ambiguity or nondeterminism than absolutely necessary for this language.

Schmidt and Szymanski showed in [11] that there is no recursive function bounding the savings achieved by general PDAs over unambiguous PDAs. More precisely, they proved that for every recursive function, there is an unambiguous language such that the difference between the size of the smallest PDA for this language and the size of the smallest unambiguous PDA cannot be bounded by this function. We then say that the tradeoff from PDA to UPDA is nonrecursive and write PDA $\xrightarrow{\text{nonrec}}$ UPDA. In fact, the general PDA used by Schmidt and Szymanski is ambiguous of degree only two, so PDA($\alpha \leqslant 2$) $\xrightarrow{\text{nonrec}}$ PDA($\alpha \leqslant 1$) holds. Borchhardt generalized this tradeoff in [1] to PDA($\alpha \leqslant k+1$) $\xrightarrow{\text{nonrec}}$ PDA($\alpha \leqslant k$) for arbitrary integers k. (He uses CFGs instead of PDAs, but this PDA-tradeoff is an immediate consequence of his CFG-tradeoff.) We extend his result to the following nonrecursive tradeoff:

Theorem 13. *For all* $k \in \mathbb{N}$,

$$\text{PDA}(\alpha \leqslant k+1, \beta \leqslant k+1) \xrightarrow{\text{nonrec}} \text{PDA}(\alpha \leqslant k).$$

Proof. For every deterministic Turing machine M that accepts after an odd number of moves when started on blank tape, we define a language A_M:

$$\begin{aligned} A_{M,0} = \{ & x_1 \# x_2^R \# x_3 \# \cdots \# x_{2n-1} \# x_{2n}^R a_1^{|x_{2n}|} a_2^{n_1} a_3^{n_2} a_4^{n_2} \ldots a_{2k+1}^{n_{k+1}} a_{2k+2}^{n_{k+1}} \mid \\ & n, n_1, \ldots, n_{k+1} \geqslant 1,\ \forall t = 1, \ldots, n-1 : x_{2t} \overset{1}{\underset{M}{\vdash}} x_{2t+1}, \\ & x_1 \text{ initial config. on blank tape, } x_{2n} \text{ final config.} \}, \end{aligned}$$

$$\begin{aligned} A_{M,i} = \{ & x_1 \# x_2^R \# x_3 \# \cdots \# x_{2n-1} \# x_{2n}^R a_1^{n_1} a_2^{n_2} a_3^{n_2} \ldots a_{2i+2}^{n_1} \ldots a_{2k+1}^{n_{k+1}} a_{2k+2}^{n_{k+1}} \mid \\ & n, n_1, \ldots, n_{k+1} \geqslant 1,\ \forall t = 1, \ldots, n : x_{2t-1} \overset{1}{\underset{M}{\vdash}} x_{2t} \} \quad \text{for } i = 1, \ldots, k, \end{aligned}$$

$$A_M = A_{M,0} \cup \cdots \cup A_{M,k}.$$

Note that the intersection of the $A_{M,i}$ contains exactly one word, which corresponds to the accepting computation of M on blank tape. For the moment assume that we already have proved the following:

1. There is a PDA for A_M with ambiguity and branching at most $k+1$ and a size recursive in the size of M.
2. There is a PDA for A_M with ambiguity and branching at most k.
3. Every PDA for A_M with ambiguity at most k must have a size at least recursive in the amount of tape used by M.

Then, if the tradeoff from PDA($\alpha \leqslant k+1, \beta \leqslant k+1$) to PDA($\alpha \leqslant k$) were recursive, there would be a recursive relation between the size of a TM and the amount of tape used by it. But then, the halting problem for this type of TM would be decidable and therefore, this tradeoff must be nonrecursive.

It remains to show the above three facts:

Proof of 1. As one can easily verify, every $A_{M,i}$ can be accepted by an unambiguous DPDA with a size recursive in the size of the TM M. The construction of these DPDAs is essentially the one described in [5, Lemma 8.6]. By unifying these DPDAs like in Theorem 9, we obtain a PDA in PDA($\alpha \leqslant k+1, \beta \leqslant k+1$) for A_M with a size recursive in the size of M.

Proof of 2. A_M can also be accepted by a PDA in PDA($\alpha \leqslant k, \beta \leqslant k$) by storing the input up to a certain length in the states and then branching only to the DPDAs for those $A_{M,i}$ to which the word with this prefix can still belong. If this length is chosen greater than the length of the one word in the intersection of the $A_{M,i}$, then this branch is at most k-fold. Since this is the only nondeterministic move in every computation and since the DPDAs for the $A_{M,i}$ are unambiguous, this PDA has ambiguity and branching at most k. Details of this construction can be found in [4].

Proof of 3. We have to show that every PDA for A_M with ambiguity at most k has a size at least recursive in N, the amount of tape used by the TM M. Since for every PDA there is an equivalent CFG with the same ambiguity and a size recursive in the size of the PDA and vice versa, it suffices to show that every CFG for A_M with ambiguity at most k has a size at least recursive in N. Now let G be some CFG with $L(G)=A_M$, $\alpha_G \leqslant k$ and let C be its pumping constant from Lemma 4. We assume G is so small that $C! + C \leqslant N$.

Let $z = z_1 \# z_2^R \# z_3 \# \cdots \# z_{2n-1} \# z_{2n}^R$ be the word corresponding to the one accepting computation of M started on blank tape and $N = |z_{2n}|$. Let

$$w_0 = z a_1^N a_2^N a_3^{N-C!} a_4^{N-C!} a_5^N \ldots a_{2k+2}^N$$

and mark all the (at least C) a_3's in this word. Then every factorization $w_0 = v_1 v_2 v_3 v_4 v_5$ satisfying (1)–(3) of Lemma 4 must have $v_2 = a_3^l$ and $v_4 = a_4^l$ for some $0 < l < C$. Setting $n = 1 + C!/l$ in (3), we get a derivation

$$S \overset{*}{\Rightarrow} v_1 A v_5 \overset{*}{\Rightarrow} v_1 v_2^n A v_4^n v_5 \overset{*}{\Rightarrow} v_1 v_2^n v_3 v_4^n v_5 = z a_1^N \ldots a_{2k+2}^N = w.$$

Now consider the words

$$w_i = z a_1^{N-C!} a_2^N \ldots a_{2i+1}^N a_{2i+2}^{N-C!} a_{2i+3}^N \ldots a_{2k+2}^N \quad \text{for } i = 1, \ldots, k,$$

where in w_i all the a_{2i+2}'s are marked. Then every factorization $w_i = v_1 v_2 v_3 v_4 v_5$ satisfying (1)–(3) of Lemma 4 must have $v_2 = a_1^l$ and $v_4 = a_{2i+2}^l$ for some $0 < l < C$. Setting $n = 1 + C!/l$ in (3), we get a derivation

$$S \overset{*}{\Rightarrow} v_1 A v_5 \overset{*}{\Rightarrow} v_1 v_2^n A v_4^n v_5 \overset{*}{\Rightarrow} v_1 v_2^n v_3 v_4^n v_5 = z a_1^N \ldots a_{2k+2}^N = w.$$

So we have $k+1$ different derivations of w, contradicting $\alpha_G \leqslant k$. Thus, $C! + C > N$ must hold. Since the pumping constant C is recursive in the size of the CFG, the size

of G must be greater than $\varphi(N)$ where φ is a fixed recursive function independent of the TM M. □

This theorem says that for certain languages which can be described by a PDA with ambiguity k, it might be preferable to describe this language by a PDA with ambiguity $k+1$ rather than k, because this description can be much smaller, although it uses more ambiguity than absolutely necessary. Alternatively, one can say that, given some PDA with finite ambiguity, it either is impossible to find an equivalent PDA with less ambiguity (which is the case for the languages A_k introduced in the last section) or if it is possible, this PDA sometimes has to be much bigger than the first PDA (which is the case for the languages A_M from above).

Another implication of this nonrecursive tradeoff is that there can be no algorithm that, given some PDA with ambiguity $k+1$ for which an equivalent PDA with ambiguity k is known to exist, actually constructs this second PDA. The reason is that the (recursive) time complexity of such an algorithm would also be a recursive upper bound for the corresponding tradeoff.

Analogously to the tradeoff PDA $\xrightarrow{\text{nonrec}}$ UPDA, Valiant showed in [12] that the tradeoff from unambiguous PDAs to deterministic PDAs is nonrecursive. Since the unambiguous PDA used in his proof has in fact only a branching of two, $\text{PDA}(\alpha \leqslant 1, \beta \leqslant 2) \xrightarrow{\text{nonrec}} \text{PDA}(\beta \leqslant 1)$ holds. We generalize this tradeoff for arbitrary integers k:

Theorem 14. *For all* $k \in \mathbb{N}$,

$$\text{PDA}(\alpha \leqslant 1, \beta \leqslant k+1) \xrightarrow{\text{nonrec}} \text{PDA}(\beta \leqslant k).$$

Proof. This proof is similar to the last one but uses a different language B_M for some given TM M:

$$B_{M,0} = \{x_1 \# x_2^R \# x_3 \# \cdots \# x_{2n-1} \# x_{2n}^R b_1^{|x_{2n}|} \mid$$
$$n \geqslant 1,\ \forall t = 1, \ldots, n-1 : x_{2t} \overset{1}{\underset{M}{\vdash}} x_{2t+1},$$
$$x_1 \text{ initial config. on blank tape, } x_{2n} \text{ final config.}\},$$

$$B_{M,i} = \{x_1 \# x_2^R \# x_3 \# \cdots \# x_{2n-1} \# x_{2n}^R b_1^m b_2^{i \cdot m} \mid$$
$$n, m \geqslant 1,\ \forall t = 1, \ldots, n : x_{2t-1} \overset{1}{\underset{M}{\vdash}} x_{2t}\} \quad \text{for } i = 1, \ldots, k,$$

$$B_M = B_{M,0} \cup \cdots \cup B_{M,k}.$$

Like in the last tradeoff, we will prove that

1. There is a PDA for B_M with ambiguity 1, branching at most $k+1$ and a size recursive in the size of M.
2. There is a PDA for B_M with ambiguity and branching at most k.

3. Every PDA for B_M with branching at most k must have a size at least recursive in the amount of tape used by M.

Then the tradeoff between PDA($\alpha \leqslant 1, \beta \leqslant k+1$) and PDA($\beta \leqslant k$) must be nonrecursive for the same reasons as in the last theorem.

Proof of 1. There is an unambiguous DPDA for every $B_{M,i}$ with a size recursive in the size of M. By unifying these DPDAs, we obtain a PDA for B_M with branching $k+1$ and a size recursive in the size of M, and since the $B_{M,i}$ are disjoint, this PDA also is unambiguous.

Proof of 2. Let $z = z_1 \# z_2^R \# z_3 \# \cdots \# z_{2n-1} \# z_{2n}^R$ be the word corresponding to the one accepting computation of the TM when started on blank tape, and let $N = |z_{2n}|$ be the amount of tape used. Then zb_1^N is the longest word which for all $B_{M,i}$ is a prefix of some word in $B_{M,i}$. Like in the last theorem, we therefore can construct a PDA for B_M with ambiguity and branching at most k.

Proof of 3. Let $M' \in \mathrm{PDA}(\beta \leqslant k)$ and $L(M') = B_M$. Due to Theorem 9 there must be k DPDAs with a size recursive in the size of M' whose union is equivalent to M'. Let C be the maximum of the pumping constants for these k DCFLs from Lemma 5. Then C is recursive in the size of the DPDAs and therefore recursive in the size of M'. Let φ be this recursive function, that is, $C = \varphi(\|M'\|)$. Now assume that M' is so small that $C = \varphi(\|M'\|) < N$ and consider the following $k+1$ words:

$$w_i = zb_1^N b_2^{iN} \quad \text{for } i = 0, \ldots, k$$

Since there are more words than DPDAs, two words w_j and $w_{j'}$, $j < j'$, must be in the same DCFL L. Like in Lemma 11 one can show that this is not possible if $j > 0$, so $j = 0$ remains. We now mark all the $N > C$ many b_1's in

$$w_0 = z_1 \# z_2^R \# z_3 \# \cdots \# z_{2n-1} \# z_{2n}^R b_1^N.$$

Then for every factorization $w_0 = v_1 v_2 v_3 v_4 v_5$ satisfying (1)–(5) of Lemma 5, v_2 is a subword of z_{2n}^R of length l ($z_{2n}^R = y v_2 y'$) and $v_4 = b_1^l$ for some $0 < l < C$. Setting $m = n = 1$ and $u = v_4 v_5 b_2^{j'N}$, we get $v_1 v_2^m v_3 u = w_{j'} \in L$ and thus,

$$v_1 v_2^{m+n} v_3 v_4^n u = z_1 \# \cdots \# z_{2n-1} \# y v_2^2 y' b_1^{N+l} b_2^{j'N} \in L.$$

Since this word contains b_2's, it is not in $B_{M,0}$. Since the TM-transition from z_{2n-1} to yv_2^2y' is incorrect due to $yv_2^2y' \neq z_{2n}^R$, this word is not in $B_{M,i}$ for any $i > 0$. Therefore, it is not in B_M, contradicting $L \subseteq B_M$. Because of this contradiction, $C = \varphi(\|M'\|) < N$ cannot hold, that is, the size of M' is at least $\varphi^{-1}(N)$. □

Analogous to the last theorem, this one shows that it might be preferable to use more nondeterminism in the description of a language than necessary, so that the description can be much smaller. Alternatively, one can say that, given some PDA with finite nondeterminism, it might be impossible to find an equivalent PDA with

less nondeterminism (which is the case for the languages B_k introduced in the last section) or if it is possible, this PDA sometimes has to be much bigger than the first PDA (which is the case for the languages B_M from above). Again, there can be no algorithm that, given some $M \in \mathrm{PDA}(\beta \leqslant k+1)$ with $L(M) \in \mathrm{CFL}(\beta \leqslant k)$, constructs an $M' \in \mathrm{PDA}(\beta \leqslant k)$ equivalent to M.

In addition to these nonrecursive tradeoffs in the last two theorems, we will show one nontrivial recursive tradeoff: We can prove that for every language the size of the smallest PDA in $\mathrm{PDA}(\alpha \leqslant k, \beta \leqslant k)$ is at most exponential in the size of the smallest PDA in $\mathrm{PDA}(\beta \leqslant k)$ for this language. We then say there is an exponential upper bound for the tradeoff between this two classes and write

$$\mathrm{PDA}(\beta \leqslant k) \xrightarrow{\leqslant 2^{cn}} \mathrm{PDA}(\alpha \leqslant k, \beta \leqslant k).$$

The proof of this exponential difference is based on the constructions in Theorem 9. There we showed that for every PDA M with $\beta_M \leqslant k$ there are k DPDAs, whose union is equivalent to M. By first making these DPDAs unambiguous and then unifying them, we get a PDA M' equivalent to M with $\beta_{M'} \leqslant k$ and $\alpha_{M'} \leqslant k$. The constructions in Theorem 9 can be implemented in such a way that the size of this resulting PDA M' and thus also the size of the smallest PDA of this type is at most exponential in the size of M.

References

[1] I. Borchhardt, Nonrecursive tradeoffs between context-free grammars with diferent constant ambiguity, Master Thesis, Universität Frankfurt/Main, 1992 (in German).

[2] J. Goldstine, C.M.R. Kintala and D. Wotschke, On measuring nondeterminism in regular languages, *Inform. Comput.* **86** (1990)179–194.

[3] M.A. Harrison, *Introduction to Formal Language Theory* (Addison-Wesley, Reading, MA, 1978).

[4] C. Herzog, Pushdown automata with bounded nondeterminism or bounded ambiguity, Master Thesis, Universität Frankfurt/Main, 1994 (in German).

[5] J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages and Computation* (Addison-Wesley, Reading, MA, 1979).

[6] C.M.R. Kintala, Refining nondeterminism in context-free languages, *Math. Systems Theory* **12** (1978) 1–8.

[7] C.M.R. Kintala and D. Wotschke, Amounts of nondeterminism in finite automata, *Acta Inform.* **13** (1980) 401–418.

[8] H. Maurer, *The Existence of Context-free Languages which are Inherently Ambiguous of any Degree*, Dept. of Mathematics Research Series, University of Calgary, 1968.

[9] K. Salomaa and S. Yu, Degrees of nondeterminism for pushdown automata, Springer Lecture Notes in Computer Science, Vol. 529 (Springer, Berlin, 1991) 380–389.

[10] K. Salomaa and S. Yu, Limited nondeterminism for pushdown automata, *EATCS Bull.* **50** (1993) 186–193.

[11] E.M. Schmidt and T.G. Szymanski, Succinctness of descriptions of unambiguous context-free languages, *SIAM J. Comput.* **6** (1977) 547–553.

[12] L. Valiant, A note on the succinctness of descriptions of deterministic languages, *Inform. and Control* **32** (1976) 139–145.

[13] D. Vermeir and W. Savitch, On the amount of nondeterminism in pushdown automata, *Fundam. Inform.* **4** (1981) 401–418.